PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/38, 15/16 — A1

(11) International Publication Number: WO 99/27460

(43) International Publication Date: 3 June 1999 (03.06.99)

(21) International Application Number: PCT/US98/23835

(22) International Filing Date: 6 November 1998 (06.11.98)

(30) Priority Data:
08/976,964 24 November 1997 (24.11.97) US

(71) Applicant *(for all designated States except US)*: POINTCAST, INC. [US/US]; 501 Macara Avenue, Sunnyvale, CA 94086 (US).

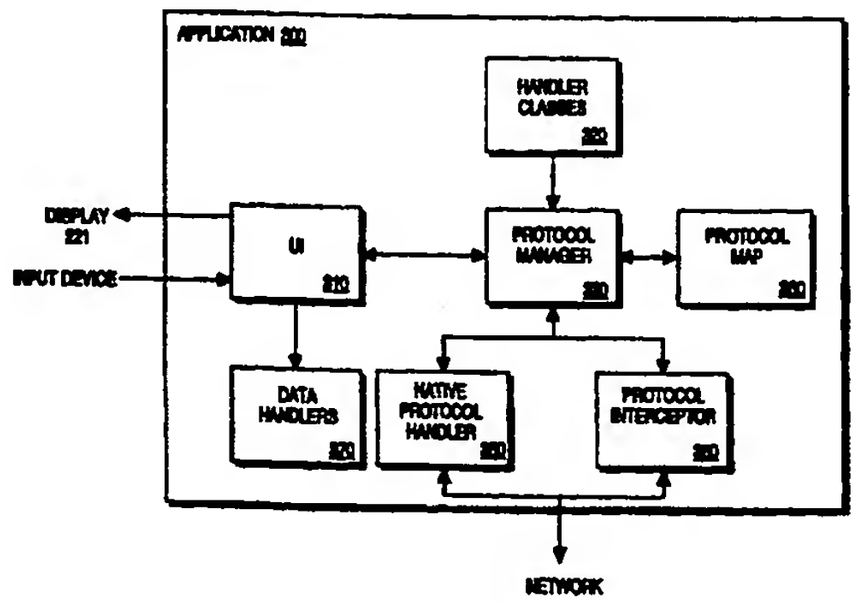
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: BELOV, Lev [RU/US]; 280 San Benito Way, San Francisco, CA 94127 (US). DOUGLAS, Jason [US/US]; 493 29th Street, San Francisco, CA 94131 (US). HASSETT, Gregory, P. [US/US]; 21925 Almaden Avenue, Cupertino, CA 95014 (US). KOVALCHUK, Aleksandr [BY/US]; 102 Locksley #106, San Francisco, CA 94122 (US). NIELSEN, Thomas, A. [US/US]; 7851 Hazelnut Drive, Newark, CA 94560 (US). RAO, Parik [US/US]; 450 North Mathilda Avenue R–103, Sunnyvale, CA 94086 (US).

(74) Agents: SALTER, James, H. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States: AL, AM, AT, AT (Utility model), AU (Petty patent), AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, EE (Utility model), ES, FI, FI (Utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: IDENTIFICATION AND PROCESSING OF COMPRESSED HYPERTEXT MARKUP LANGUAGE (HTML)



(57) Abstract

A mechanism for recognizing and processing compressed hypertext markup language (HTML) data is provided. According to one aspect of the present invention, compressed HTML is downloaded by an application by employing a protocol interceptor (350), such as an instance of a pluggable protocol handler (340) (also referred to as a moniker), a plug–in, a helper application, or the like. The application (300) receives a request for a resource located on a remote server (120) that contains compressed HTML data. The application (300) invokes an appropriate protocol interceptor (350) to downloa d the requested resource. The appropriate protocol interceptor (350) may be determined with reference to a protocol map (360) containing a registered asynchronous pluggable protocol, pluggable name–space handlers, and/or MIME filters. In anyevent, the protocol interceptor (350) requests the resource from the remote serverrequests the resource from the remote server (120) by way of a network transfer protocol, such as Hypertext Tran sfer Protocol (HTTP), File Transfer Protocol (FTP), gopher, Simple Mail Transfer Protocol (SMTP), or similar application protocol. Compressed HTML data is subsequently received by the protocol interceptor (350) from the remote server. The protocol interceptor (350) decompresses the compressed HTML data and provides HTML data to the application (300) which may then be rendered by the application.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

IDENTIFICATION AND PROCESSING OF COMPRESSED HYPERTEXT MARKUP LANGUAGE (HTML)

FIELD OF THE INVENTION

The invention relates generally to the field of Web-based applications. More particularly, the invention relates to processing HTML data that is stored on a server in a compressed format.

BACKGROUND OF THE INVENTION

The use of the Internet, and particularly the World Wide Web (the Web) has increased substantially in recent years. The Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. Although the Web has in the past been a source of primarily scientific information, it is now a valuable resource for information relating to almost any subject, including business, entertainment, travel, and education, to name just a few.

The architecture of the Web follows a conventional client-server model. The terms "client" and "server" are used to refer to a computer's general role as a requester of data (the client) or provider of data (the server). Web clients and Web servers communicate using a protocol such as Hypertext Transfer Protocol (HTTP). HTTP is an application-level network transfer protocol that, among other things, facilitates the request and the transfer of data between Web clients and Web servers.

In the Web environment, Web browsers, programs that can download and render Web documents, reside on clients and the Web documents (also referred to as Web pages) reside on servers. Web documents are typically encoded with a tag-based notation language called Hypertext Markup Language (HTML).

Many different file formats are encountered on the Web such as HTML files, plain text files, word processing documents, graphics, video, audio, software applications, and many others. Currently, a significant portion of the content of

the Web resides on servers, and is consequently transmitted to clients, in an uncompressed format. While many graphic and audio file formats are typically compressed, such as Graphics Interchange Format (GIF) and Real Audio files, HTML files are not. In addition to the clear waste of disk space on servers, many downstream inefficiencies are a result of uncompressed HTML data transmission, including the depletion of intangible resources such as Internet bandwidth and user time.

This oversight regarding HTML files may be due in part to the fact that no standardized mechanism has been agreed upon by content providers for identifying compressed content as such. Additionally, the focus of traditional solutions is directed toward providing hardware that can process data more quickly, such as faster modems and processors.

However, now that next generation Web browsers, such as Microsoft® Internet Explorer™ 4.0, are providing pluggable protocols and other mechanisms for registering customized Uniform Resource Locator (URL) protocol handlers, it is desirable to exploit these tools to allow servers to store and transmit compressed HTML. Additionally, it is advantageous for client programs to recognize compressed content, and to perform appropriate processing including client-side decompression. Further, in view of the anticipated pervasiveness of push technology, efficient handling and recognition of compressed HTML is especially important.

SUMMARY OF THE INVENTION

A mechanism for recognizing and processing compressed hypertext markup language (HTML) data is described. According to one aspect of the present invention, compressed HTML is downloaded by an application by employing a protocol interceptor.

The application receives a request for a resource located on a remote server that contains compressed HTML data. The application invokes an appropriate protocol interceptor to download the requested resource. The protocol interceptor

requests the resource from the remote server by way of a network transfer protocol. Compressed HTML data is received by the protocol interceptor. The protocol interceptor decompresses the compressed HTML data and provides HTML data to the application which may then be rendered by the application. By employing a protocol interceptor as an intermediary between a Web server and a client-side application, HTML data may be transferred in a compressed format from the Web server to the client transparently to the requesting application. Advantageously, transferring HTML data in compressed format conserves network bandwidth and decreases the amount of time required to download Web documents.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is a logical view of a client/server network.

Figure 2 is an example of a computer system upon which one embodiment of the present invention can be implemented.

Figure 3 is a block diagram illustrating components of an exemplary application according to one embodiment of the present invention.

Figure 4 is a flow diagram illustrating resource request processing according to one embodiment of the present invention.

DETAILED DESCRIPTION

A mechanism for recognizing and processing compressed HTML data is described. According to the present invention, HTML data may be transferred in a compressed format from a Web server to a client transparently to the requesting

client application by using a protocol interceptor as an intermediary between the Web server and the application. For example, when the application encounters a resource that satisfies a predetermined condition, the protocol interceptor may be called upon to transfer the data from the Web server to the client. The protocol interceptor uses an appropriate protocol to download the compressed data stream and then transforms the compressed data stream into an HTML data stream that is recognizable to the application. In this manner, the application need not have knowledge regarding the data's intermediate storage and transmission format. Rather, the application can simply call upon a protocol interceptor that is associated with the resource and receive HTML data.

In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form.

The present invention includes various steps, which will be described below. The steps of the present invention may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps.

Importantly, while embodiments of the present invention are described with reference to a particular mechanism for recognizing compressed content such as inspecting the scheme of a requested URL, for example, those of ordinary skill in the art will be aware of equivalent mechanisms. Additionally, embodiments of the present invention are described with reference to a particular mechanism for processing compressed HTML, such as a moniker. However, alternative mechanisms such as plug-ins, helper applications and the like are contemplated. Finally, while the present invention may be embodied in a Web browser such as Microsoft Internet Explorer, the method described herein is

equally applicable to other Web-based applications and browsers such as Netscape Navigator™, Lotus Notes™, and others.

Before describing the novel method of recognizing and processing compressed HTML, the client-server model as applied to the Web and the conventional method of transmitting and rendering Web documents will briefly be discussed with reference to Figure 1. Figure 1 depicts a plurality of clients 110 and servers 120 coupled in communication with a network 100. Network 100 may represent the infrastructure of a Local Area Network (LAN), an Intranet comprising a collection of LANs, a Wide Area Network (WAN) spanning geographical areas, or the global web of interconnected computers and computer networks that make up the Internet, for example. An exemplary client 110 architecture is discussed below.

The process of serving a Web document to an Internet user can be described in three steps. The Web browser issues a request for a Web document, or more generally, a "resource" on a specific Web server 120. This request is typically in the form of a HTTP Get request which provides the server 120 with a logical address, such as a Uniform Resource Locator (URL), of the requested Web document. When the server 120 receives the request, it searches its document space for the requested Web document, and transmits the Web document in an uncompressed HTML format to the Web browser. Upon receipt, the Web browser renders the Web document as directed by the HTML encoding contained therein.

AN EXEMPLARY CLIENT SYSTEM

Referring to Figure 2, a computer system is shown as 200. The computer system 200 represents an exemplary client 110 upon which one embodiment of the present invention may be implemented. Computer system 200 comprises a bus or other communication means 201 for communicating information, and a processing means such as processor 202 coupled with bus 201 for processing information. Computer system 200 further comprises a random access memory (RAM) or other

dynamic storage device 204 (referred to as main memory), coupled to bus 201 for storing information and instructions to be executed by processor 202. Main memory 204 also may be used for storing temporary variables or other intermediate information during execution of instructions by processor 202. Computer system 200 also comprises a read only memory (ROM) and/or other static storage device 206 coupled to bus 201 for storing static information and instructions for processor 202. Data storage device 207 such as a magnetic disk or optical disc and its corresponding drive may be coupled to bus 201 for storing information and instructions.

Computer system 200 can also be coupled via bus 201 to a display device 221, such as a cathode ray tube (CRT), for displaying information to a computer user. For example, Web documents, graphics, video clips, and other data types may be presented to the user on the display device 221. Typically, an alphanumeric input device 222, including alphanumeric and other keys, is coupled to bus 201 for communicating information and/or command selections to processor 202. Another type of user input device is cursor control 223, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 202 and for controlling cursor movement on display 221.

A communication device 225 is also coupled to bus 201 for accessing servers via the Internet, for example. The communication device 225 may include a modem, a network interface card, or other well known interface devices such as those used for coupling to an Ethernet, token ring, or other types of networks. In any event, in this manner, the computer system 200 may be coupled to a number of servers via a conventional network infrastructure, such as a company's Intranet and/or the Internet, for example.

AN EXEMPLARY APPLICATION

Figure 3 illustrates a block diagram of an exemplary application 300. In the embodiment depicted, the application 300 may be a Web-based application

such as Microsoft Internet Explorer or similar Web browser capable of requesting Web documents from servers via HTTP and rendering those Web documents.

In the embodiment depicted, the application 300 includes a user interface (UI) 310, a protocol manager 330, a set of handler classes 320, a protocol map 360, a set of data handlers 370, a set of native protocol handlers 340, and a protocol interceptor 350. Each of these items will briefly be described below.

A user may provide input to the application 300, such as an indication of a link or URL to which the user desires to navigate through UI 310. The UI 310 may additionally provide visual feedback to the user via the display 221 in the form of text and/or graphic images representing Web documents corresponding to resources requested by the user.

Resource requests from the user are processed by the protocol manager 330. In one embodiment, the protocol manager 330 determines with reference to the protocol map 360 whether or not the requested resource is associated with a natively supported protocol such as HTTP.

If the requested resource is associated with a protocol that is natively supported by the application 300, then the resource may be downloaded to the client upon which the application is executing by a protocol handler from the set of native protocol handler objects 340. The natively supported protocols may include standard Internet protocols that the application 300 itself knows how to process. The protocol associated with a particular resource request refers to the predefined application-level communications for downloading the resource to the client from the remove server. For example, the protocol may specify how requests and acknowledgments are handled and the structure of communicated data. At any rate, Web browsers typically natively support HTTP, File Transfer Protocol (FTP), Gopher, and other application protocols. Typically, if a particular protocol becomes prevalent enough on the Web, later generations of Web browsers can be expected to provide native support for such protocol. Therefore, it should be appreciated that more or less native protocol handlers may be employed in alternative embodiments.

If the protocol manager 330 determines no natively supported protocol is associated with the resource request, then an appropriate non-native handler from handler classes 320 such as a protocol interceptor 350 may be employed. There are many ways to implement the protocol interceptor 350, some of which will be discussed below. According to this embodiment, handler classes 320 may include one or more classes of protocol handlers for protocols that are not supported natively by the application 300. For purposes of discussion herein, a "class" can be thought of as a piece of executable code while an "object" refers to a particular instance of a class.

To facilitate the location of an appropriate handler class corresponding to a particular protocol, a mapping of protocols to handler classes may be provided by a protocol map 360. Alternatively, the mapping may use one or more portions of the URL to map to particular handler classes. At any rate, in this manner, when no native support is provided for a particular protocol, the application 300 may instantiate (e.g., make an instance of) an appropriate handler from the set of handler classes 320 with reference to the protocol map 360, for example. In one embodiment, the protocol map 360 may be implemented with the Windows™ Registry. However, the protocol map 360 is not limited to such a file. All that is needed is a mechanism for associating protocols or groups of resources with a set of instructions such as a handler for processing data according to the corresponding protocol. Processing, in this context, typically involves downloading an input data stream from a server and transforming the input data stream from its original data type into a data stream that is recognizable by the application 300, if necessary. In alternative embodiments, this association/mapping may be maintained in a local database or any other file so long as the protocol manager 330 is configured appropriately so as to find it.

When the UI 310 receives the requested resource from the protocol manager 330, for purposes rendering the Web document, the application 300 may employ the data handlers 370. Web browsers typically natively support Graphics Interchange Format (GIF) and Joint Photographic Experts Group (JPEG) graphic

files which are commonly used for inline images. The data handlers 370 may include one or more data handlers for rendering HTML, GIF, and/or JPEG files, for example.

UNIFORM RESOURCE LOCATOR (URL) SYNTAX

Before describing some exemplary handler classes, it may be helpful at this point to briefly describe the syntax of a URL. URLs follow the syntax described in Request for Comments (RFC) 1738, Uniform Resource Locators (URL), December 1994. According to RFC 1738, a URL contains the name of the "scheme" being used (e.g., http, ftp, gopher, etc.) followed by a colon and then a string, the "scheme-specific part" whose interpretation depends on the scheme. URLs are, therefore, written as follows:

<scheme>:<scheme-specific part>

For example, the PointCast, Inc. Web site is located at the following URL: "http://www.pointcast.com". The scheme is "http" and the scheme-specific part is "www.pointcast.com".

Resources having the same data type may be identified by grouping resources of a particular data type within a common scheme. For example, according to one embodiment of the present invention, the scheme "pcn" (short for PointCast Network) may be used to identify the compressed HTML data type. Thus, the file identified by the URL "pcn:/<path>/<filename>", for example, would be known by the protocol interceptor 350 to contain compressed HTML simply by referring to the scheme of the URL.

Assuming the naming convention of "pcn" for the scheme of compressed HTML has been established, an appropriate handler class that knows how to communicate with the server upon which these files are located may be added to the handler classes 320 and registered with the protocol map 360 to be associated with the "pcn" scheme. Subsequently, requests received by the application 300 for resources associated with the "pcn" scheme will be routed to the appropriate handler by the application 300. The handler will be responsible for downloading

the requested file, decompressing the compressed HTML data stream, and providing an HTML data stream to the application 300. In this example, the choice of "pcn" is an arbitrary one, therefore it should be appreciated that other schemes may be employed.

Additional mappings that are contemplated may employ a standardized set of types such as Multipurpose Internet Mail Extensions (MIME) types. Alternatively, resources having the same data type may be identified with reference to the scheme-specific part of the URL such as the file extension. For example, a ".pcn" extension may be useful for identifying a compressed HTML file.

It is worth mentioning that the particular compression algorithm employed on the servers 120 for producing compressed HTML files is unimportant so long as the clients 110 use a corresponding decompression algorithm.

EXEMPLARY HANDLER CLASSES

Many possibilities exist for implementing the handler classes of the present invention for purposes of recognizing and processing compressed HTML. For example, helper applications, plug-ins, full protocol handlers, monikers or the like may be employed. Helper applications (also referred to as external viewers) are programs external to the Web browser that are used to render data types that the browser doesn't recognize natively. Helper applications typically display the file separately from the Web browser.

Plug-ins are software programs used in conjunction with Web browsers that are responsible for both data delivery (download) and display.

Full protocol handlers are client-side objects instantiated by Web-enabled applications to allow the transparent delivery of Internet content. Similarly, a moniker is a client-side object that encapsulates a data item's name, location, and the operations required to open it. For example, a moniker might contain a file name or a URL. A moniker may be thought of as a protocol handler that gets in-process handling rather than launching a separate process. Monikers support an

operation known as binding, which is the process of locating the object named by the moniker, activating it or loading it in memory if it isn't already there, and returning an interface pointer to it.

To facilitate the routing of resource requests to custom URL protocol handlers, it is desirable to form an association between the resource and the handler class. In Microsoft® Internet Explorer™ a mechanism is provided for defining new protocols with the Asynchronous Pluggable Protocols Application Program Interface (API), which allows specific protocol schemes, for example, to be mapped to a handler class. This type of mapping may be accomplished in a number of ways including registration of an asynchronous pluggable protocol, registration of a pluggable name-space handler, and registration of a MIME filter.

An asynchronous pluggable protocol (handler class) may be registered in the protocol map 360 (e.g., Windows Registry). Asynchronous pluggable protocols allow attempts to navigate to URLs with a specified scheme, such as http, ftp, and the like, to be routed to a custom URL protocol handler registered in the protocol map 360 (e.g., a system registry, Windows Registry, or the like). Therefore, after the handler has been registered, the handler will be used for any URLs containing the specified scheme.

A pluggable name-space handler can be set up to handle one or more specific patterns for a given URL scheme. When one of the specified patterns is found in a URL scheme-specific part of the appropriate scheme, the handler is used rather than the default protocol for the scheme. A pluggable name-space handler may be used to identify compressed HTML files having a predetermined file extension, or key phrase in the path, for example.

A MIME filter may be registered for a particular pattern in the MIME content-type header field. The handler may be used to manipulate the compressed HTML data stream it receives and return a decompressed HTML data stream for received MIME messages having the specified content-type.

RESOURCE REQUEST PROCESSING

Resource request processing will now be described with reference to Figure 4. According to the embodiment depicted, the flow of the resource request processing is determined based upon the resource request itself. Several mechanisms for determining the appropriate protocol handler have been discussed, for example, the appropriate protocol handler may be determined with reference to the scheme of the URL or the extension of the file being requested. At step 405, a resource request (e.g., a request for a file, a database record, etc.), typically in the form of a URL including zero or more parameters, is received by the application 300 in the UI 310, for example.

At step 410, as described above, subsequent processing may be determined based upon the resource request (e.g., the URL). The appropriate protocol handler for downloading the requested resource to the client may be determined with reference to the scheme of the URL, the MIME type associated with the server's response, or other similar mechanisms, for example. According to one embodiment, the appropriate handler is determined by searching the protocol map 360 for a scheme that matches the scheme of the URL of step 405. In alternative embodiments, a mapping of file extensions or MIME types to handler classes may be maintained in the protocol map 360, as described above.

Those of ordinary skill in the art will appreciate, in alternative embodiments, such as those employing plug-ins, helper applications, or equivalent software programs, rather than searching the protocol map 360, the protocol manager 330 may look to a local directory where it expects to find an appropriate program to handle the resource request. In any event, according to the embodiment depicted, if the protocol handler corresponds to a native protocol handler then processing continues with step 415. Otherwise, if the protocol handler is one for processing compressed HTML, then processing continues with step 425.

At step 415, native protocol handler processing is performed, as described above.

At step 425, the appropriate handler, identified at step 410, is instantiated. Preferably, the handler object, such as protocol interceptor 350 is an asynchronous task which allows the application 300 to continue other processing while downloading of the requested resource proceeds. This objective may be met by employing a URL moniker, an asynchronous moniker that may return from a binding call before the bound object is completely ready. In this manner, execution of application 300 may continue while the object binding completes. A URL moniker allows a client program such as application 300 to create a moniker whose data is referenced by a URL. Alternatively, the handler may be implemented as an asynchronous pluggable protocol. As described above, asynchronous pluggable protocols allow attempts to navigate to URLs with a specified scheme, to be routed to a custom URL protocol handler.

At step 430, the resource request is passed to the handler. Alternatively, this may be performed when the handler is instantiated in step 425. At step 435, the handler determines if the requested resource is present in a local cache. If so, assuming the resource is unchanged at the source, at step 440, the resource may be retrieved from the cache rather than having to make a request to a remote server, for example.

At step 445, the handler transmits a request to the remote server, such as a HTTP request, corresponding to the resource request. At step 450, HTTP response data is received in the form of compressed HTML.

At step 455, the handler decompresses the data received in response to the request of step 445. Importantly, it may take more than one HTTP response to receive all the data corresponding the HTTP request. If this is the case, then decompression may be performed upon receipt of each subsequent HTTP response.

At step 460, the decompressed HTML data is provided to the application 300. According to one embodiment, the protocol interceptor 350 supports streaming of the decompressed HTML data. That is, the protocol interceptor 350 allows the application 300 to begin rendering the HTML data before the entire

file and/or any embedded content have been completely downloaded and decompressed. Since rendering of the HTML data can begin earlier, streaming has the advantage of reducing the perceived download time associated with viewing Web documents. In other embodiments, the protocol interceptor 350 may simply provide the decompressed HTML data to the application 300 when downloading and decompression are complete.

At step 465, the application 300 renders the HTML data in a well known manner using one of the native data handlers 370, for example.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method of rendering compressed hypertext markup language (HTML), the method comprising the steps of:
an application receiving a request for a resource located on a remote server that contains compressed HTML data;
the application downloading the resource by invoking a protocol interceptor;
the protocol interceptor requesting the resource from the remote server by way of a network transfer protocol;
the protocol interceptor receiving compressed HTML data;
the protocol interceptor providing HTML data to the application for display by decompressing the compressed HTML data; and
the application rendering the HTML data.

2. The method of claim 1, further including the step of selecting the protocol interceptor from a group of one or more protocol interceptors.

3. The method of claim 2, wherein prior to the step of selecting the protocol interceptor from a group of one or more protocol interceptors, the application performs the step of determining whether the resource is associated with a natively recognized protocol.

4. The method of claim 2, wherein the request for the resource comprises a Uniform Resource Locator (URL), the URL including a name of a scheme and a scheme-specific part.

5. The method of claim 4, wherein the step of selecting the protocol interceptor from a group of one or more protocol interceptors is based upon the scheme.

6. The method of claim 5, further including the step of registering the protocol interceptor as an Asynchronous Pluggable Protocol.

7. The method of claim 5, wherein the network transfer protocol comprises Hypertext Transfer Protocol (HTTP).

8. The method of claim 5, wherein the network transfer protocol comprises File Transfer Protocol (FTP).

9. The method of claim 5, wherein the network transfer protocol comprises Gopher.

10. The method of claim 5, wherein the network transfer protocol comprises Simple Mail Transfer Protocol (SMTP).

11. The method of claim 4, wherein the step of selecting the protocol interceptor from a group of one or more protocol interceptors is based upon the scheme-specific part of the URL.

12. The method of claim 11, further including the step of registering the protocol interceptor as a Pluggable Name-Space Handler.

13. A computer system comprising:

a storage device having stored therein a protocol interceptor for downloading resources having contained therein compressed HTML data; and

a processor coupled to the storage device for executing the protocol interceptor to download a resource located on a remote server, and to generate HTML data, where:

the resource located on the remote server is downloaded from the remote server by a network transfer protocol, and

the HTML data is generated by decompressing the compressed HTML data in the resource.

14. A machine-readable medium having stored thereon data representing sequences of instructions, said sequences of instructions which, when executed by a processor, cause said processor to perform the steps of:
receiving a request for a resource located on a remote server that contains compressed hypertext markup language (HTML) data;
downloading the resource by requesting the resource from the remote server by way of a network transfer protocol;
receiving compressed HTML data;
providing HTML data by decompressing the compressed HTML data; and
rendering the HTML data.

15. The machine-readable medium of claim 14, wherein said sequences of instructions when executed by a processor further cause said processor to perform the step of selecting a protocol interceptor from a group of one or more protocol interceptors.

16. The machine-readable medium of claim 15, wherein prior to the step of selecting a protocol interceptor from a group of one or more protocol interceptors, said sequences of instructions causing said processor to perform the step of determining whether the resource is associated with a natively recognized protocol.

17. The machine-readable medium of claim 15, wherein the request for the resource comprises a Uniform Resource Locator (URL), the URL including a name of a scheme and a scheme-specific part.

18. The machine-readable medium of claim 17, wherein the step of selecting a protocol interceptor from a group of one or more protocol interceptors is based upon the scheme.

19. The machine-readable medium of claim 18, wherein the network transfer protocol comprises Hypertext Transfer Protocol (HTTP).

20. The machine-readable medium of claim 17, wherein the step of selecting a protocol interceptor from a group of one or more protocol interceptors is based upon the scheme-specific part of the URL.

21. A method of rendering compressed hypertext markup language (HTML), the method comprising the steps of:
providing a protocol interceptor for use with resources associated with a scheme, the scheme indicating the resource includes compressed HTML data;
an application receiving a request for a resource associated with the scheme;
the application invoking the protocol interceptor with a resource locator identifying the resource;
the protocol interceptor requesting the resource from a source identified by the resource locator by way of a network transfer protocol;
the protocol interceptor receiving compressed HTML data;
the protocol interceptor providing HTML data to the application for display by decompressing the compressed HTML data; and
the application rendering the HTML data.

22. The method of claim 21, the method further including the step of the application determining the resource is not associated with a natively recognized protocol based upon the scheme.

23. The method of claim 22, wherein the step of the application determining the resource is not associated with a natively recognized protocol based upon the scheme further includes the step of searching a protocol map.

24. The method of claim 21, further including the step of selecting the protocol interceptor from a group of one or more protocol interceptors.

25. A method of providing hypertext markup language (HTML) data to an application, the method comprising the steps of:

receiving a request from an application for a resource, the request being accompanied by a resource locator;

requesting the resource from the source identified by the resource locator by way of a network transfer protocol;

receiving a compressed steam of data;

generating HTML data by decompressing the compressed stream of data; and

streaming the HTML data to the application for display.

26. The method of claim 25, wherein the application is a Web-based application.

27. The method of claim 25, wherein the application is a Web browser.

28. The method of claim 25, wherein the network transfer protocol comprises hypertext transfer protocol (HTTP).

29. The method of claim 25, wherein the resource comprises an Internet resource.

30. The method of claim 25, wherein the resource comprises an Intranet resource.

31. The method of claim 25, wherein the protocol interceptor comprises a moniker.

32. The method of claim 25, wherein the protocol interceptor comprises a helper application.

33. The method of claim 25, wherein the protocol interceptor comprises a plug-in.

34. The method of claim 25, wherein the protocol interceptor comprises an asynchronous pluggable protocol handler.

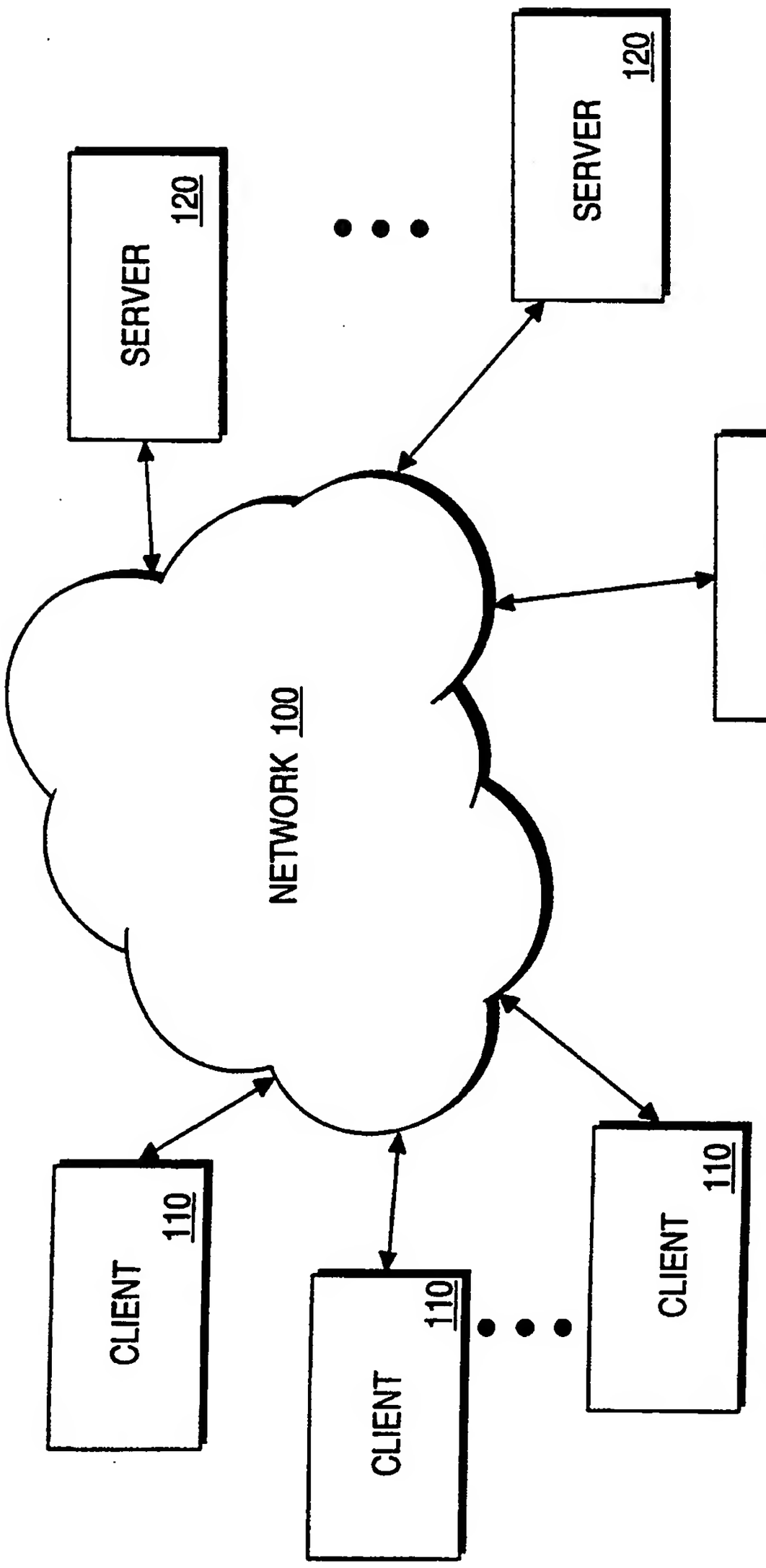

SERVER 120
SERVER 120
SERVER 120
NETWORK 100
CLIENT 110
CLIENT 110
CLIENT 110

Fig. 1

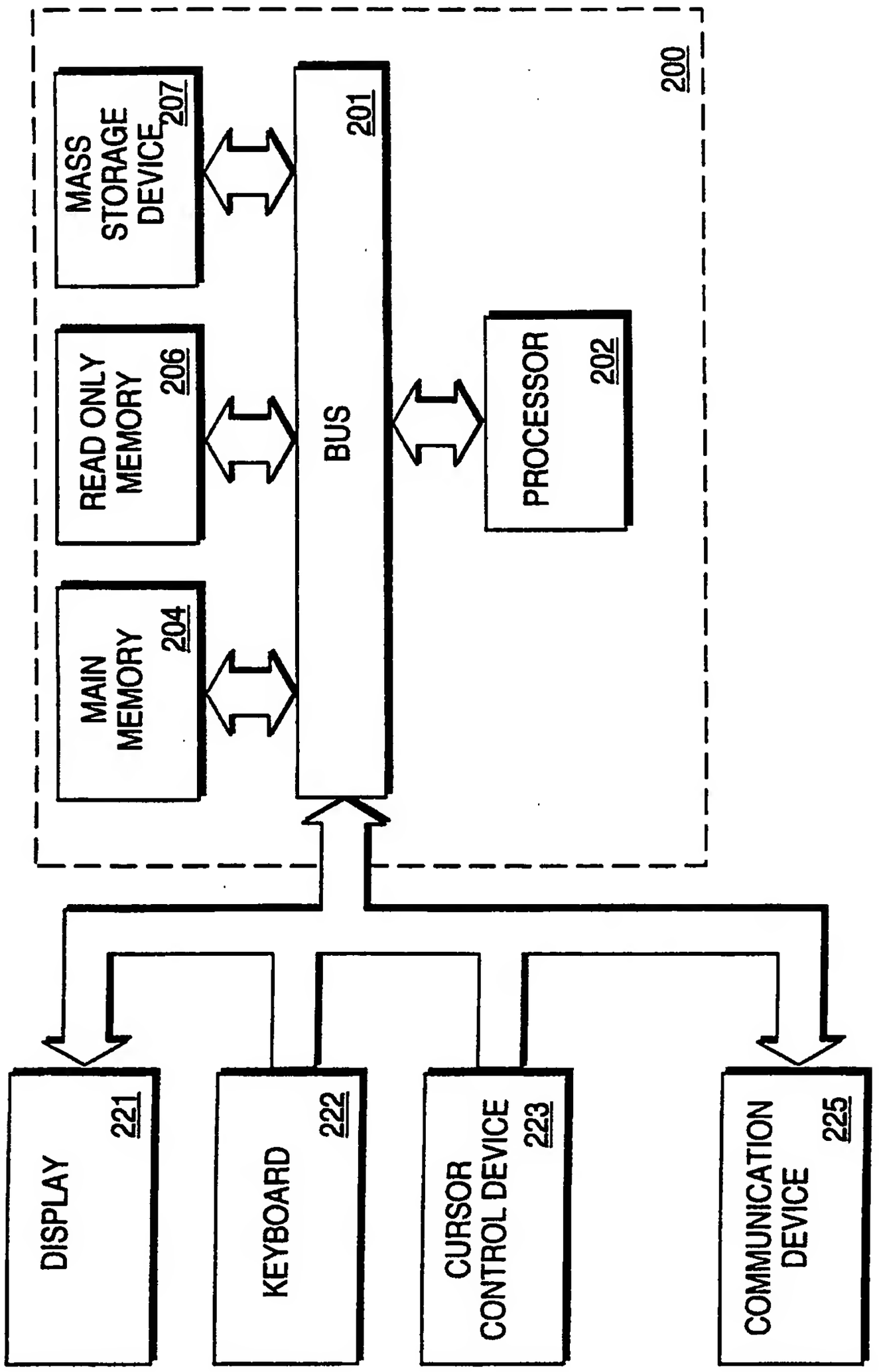

200
MASS STORAGE DEVICE 207
READ ONLY MEMORY 206
MAIN MEMORY 204
BUS
201
PROCESSOR 202
DISPLAY 221
KEYBOARD 222
CURSOR CONTROL DEVICE 223
COMMUNICATION DEVICE 225

Fig. 2

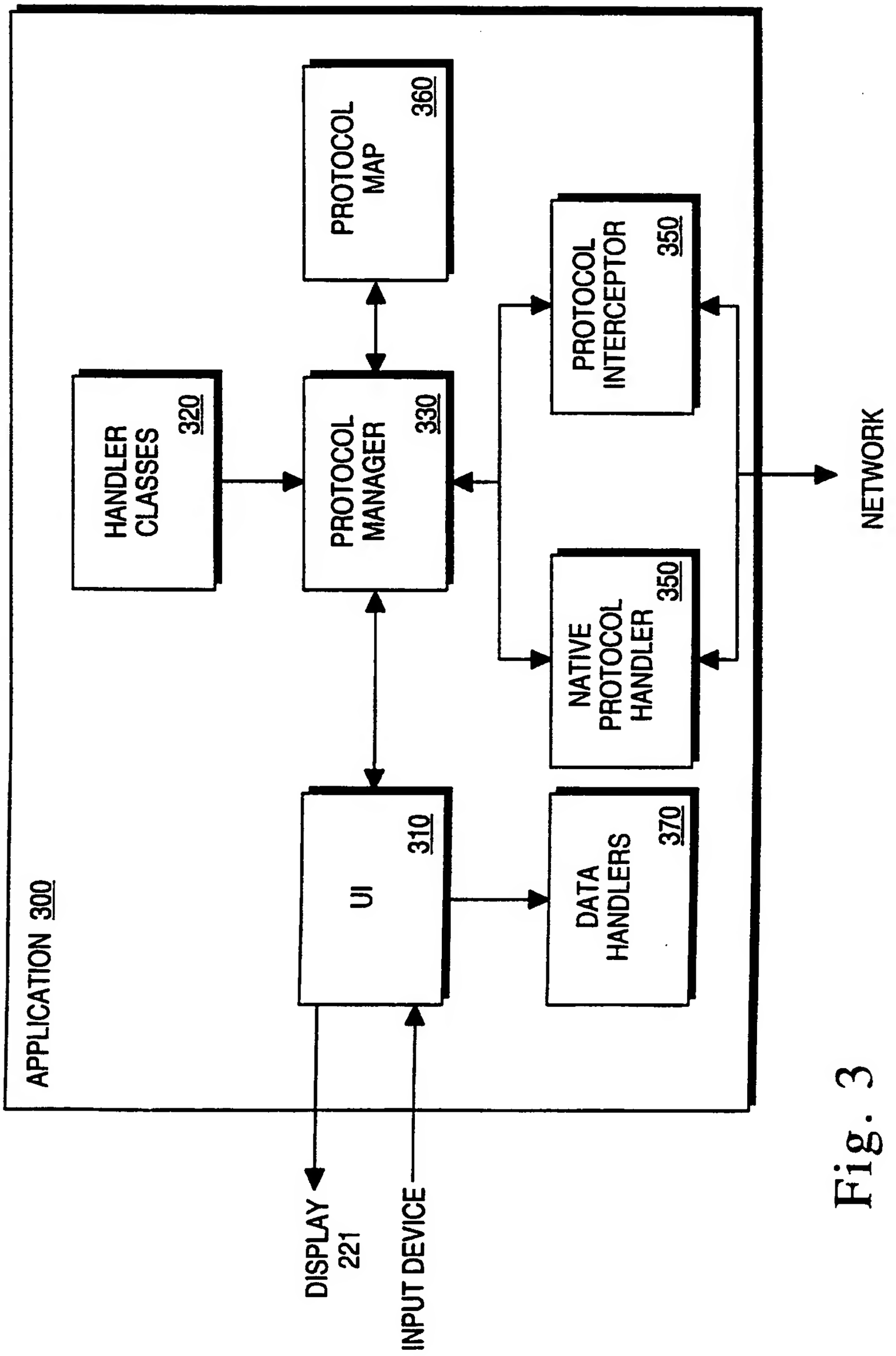

APPLICATION 300
HANDLER CLASSES 320
PROTOCOL MANAGER 330
PROTOCOL MAP 360
PROTOCOL INTERCEPTOR 350
NATIVE PROTOCOL HANDLER 350
UI 310
DATA HANDLERS 370
NETWORK
DISPLAY 221
INPUT DEVICE

Fig. 3

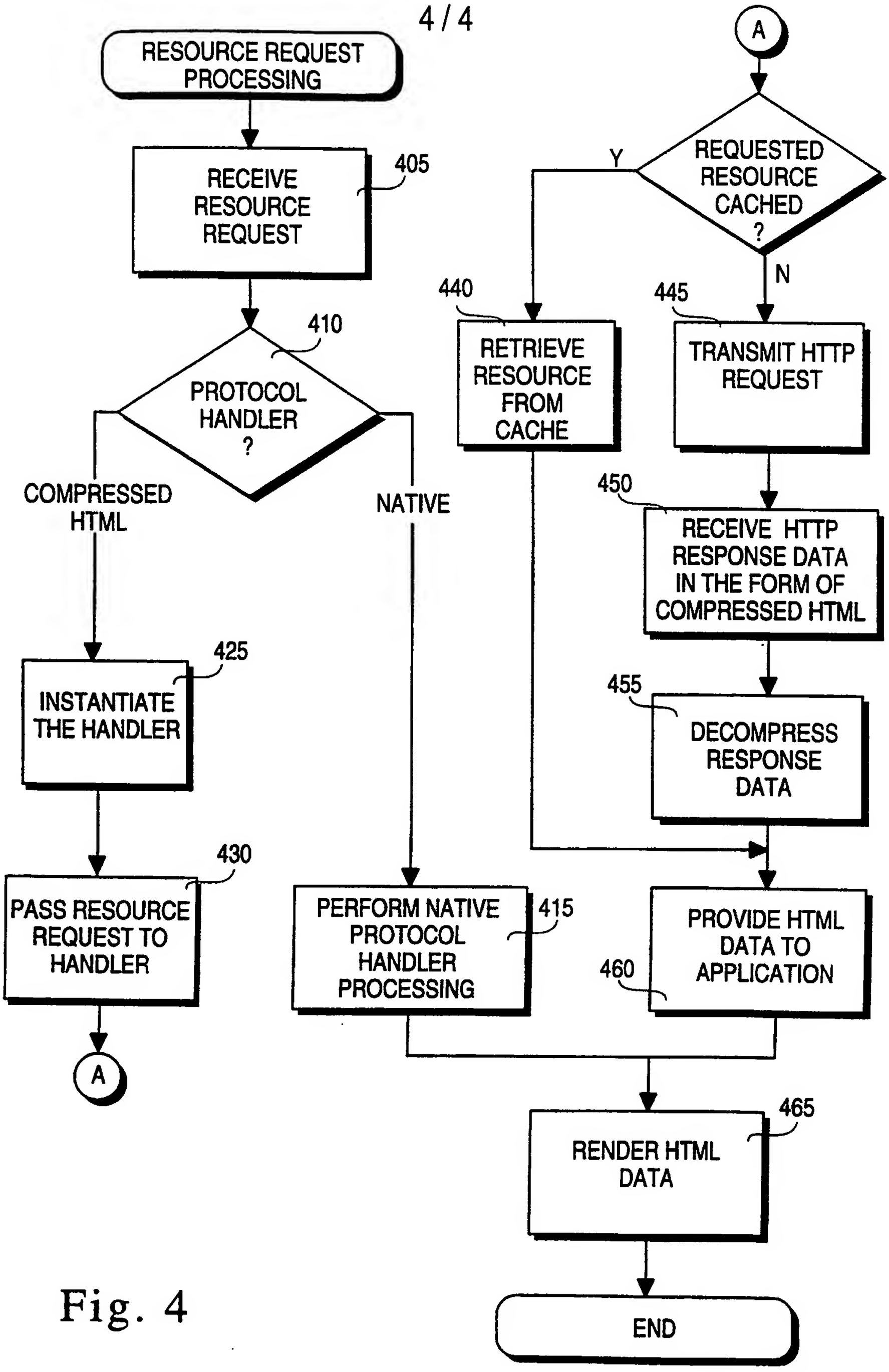

RESOURCE REQUEST PROCESSING
RECEIVE RESOURCE REQUEST
405
PROTOCOL HANDLER ?
410
COMPRESSED HTML
NATIVE
INSTANTIATE THE HANDLER
425
PASS RESOURCE REQUEST TO HANDLER
430
A
A
REQUESTED RESOURCE CACHED ?
Y
N
440
RETRIEVE RESOURCE FROM CACHE
445
TRANSMIT HTTP REQUEST
450
RECEIVE HTTP RESPONSE DATA IN THE FORM OF COMPRESSED HTML
455
DECOMPRESS RESPONSE DATA
PERFORM NATIVE PROTOCOL HANDLER PROCESSING
415
460
PROVIDE HTML DATA TO APPLICATION
465
RENDER HTML DATA
END

Fig. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/23835

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06F 13/38 , 15/16
US CL : 395/200.77, 200.12
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/200.77, 200.12

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US 5,838,927 A (GILLON et al) 17 November 1998, abstract, col. 3-col. 4, lines 1 -68.	1-34
Y,P	US 5,706,437 A (KIRCHNER et al) 06 January 1998, col. 5, lines 1-4.	1-34
Y	Metz, Cade Netscape Navigator, Personal Edition PC Magazine, v 14, n17, p188(1), October 1995.	1-34
Y	Adam Gaffin NetworkWorld Review, Networkworld Test Alliance, page 57, May 1995.	1-34

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
21 JANUARY 1999	22 MAR 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 308-3357	Authorized officer HIEU C. LE James R. Matthews Telephone No. (703) 306-3101

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/23835

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Metz Cade, Internet and online. (Netscape Navigator; America online; CompuServe/Internet Division Internet Office) PC magazine, vol15, n1, p114(2), January 1996.	1-34
Y	Roti, Steve a look Progress. (Process Soltware 's Progess 6.0 relational data base management solfware), DBMS V6, N6, P85(2), June 1993.	1-34